NFORMATION AND COMPUTATION 74, 91–112 (1987)

Identification of Pattern Languages from Examples and Queries*

ASSAF MARRON

Department of Computer Science, University of Houston, Houston, Texas 77004

AND

KER-I KO

Department of Computer Science, SUNY at Stony Brook, Stony Brook, New York 11794

Patterns are words over an alphabet of constants and variables. New words are created from a pattern as sstrings of constants are substituted for the variables of the pattern. In this paper we investigate the inductive inference of patterns from positive data and quaries, from a complexity-theoretic point of view. Using results from combinatorics on words, we give simple but nontrivial sufficient conditions on the set of the initial examples that guarantee the identification of a unique pattern by making only polynomially many quaries. Counterexamples are also provided to show that the conditions are necessary. © 1987 Academic Press, Inc.

1. INTRODUCTION

One of the typical problems addressed in the theory of inductive inference is the problem of identifying formal languages from partial information. There are, in general, two types of inference methods, depending upon whether the language is presented by examples or by queries (Gold, 1967; Angluin and Smith, 1983). A partial presentation of a language L by examples is simply two disjoint finite sets $S \subseteq L$ and $T \subseteq \bar{L}$. (A special case of this presentation, called a positive presentation, is when T is empty.) The inference problem associated with this presentation is to find a formal description D (e.g., an acceptor) of a language L' such that $S \subseteq L'$ and $T \subseteq \bar{L}'$, and D satisfies certain restrictions (e.g., D is the *smallest* acceptor among all candidates.) For example, the problem of finding, from given finite sets S and T of strings, a smallest deterministic finite automaton that

* This research was supported in part by the National Science Foundation under Grants MCS-8103479A01 and DCR-8696135. Part of the work was done while the second author was at the Department of Computer Science, University of Houston, Houston, Texas.

0890-5401/87 $3.00

recognizes a language L such that $S \subseteq L$ and $\subseteq \bar{L}$, has been studied by Gold (1978) and Angluin (1978). This problem has been shown to be NP-hard. An inference method is said to *identify L in the limit* if for larger and larger collections of examples $S \subseteq L$ and $T \subseteq \bar{L}$, the descriptions D eventually converge to a correct description for L.

The presentation of a language L by queries is an oracle that answers, for any query s, whether $s \in L$ or not. The inference method for this presentation is an oracle machine that outputs a description of L after making some queries about L. In practice, to provide the oracle machine with some starting point, examples may also be given and thus provide a mixed presentation by both examples and queries. Gold (1972), Pao and Carr (1978), and Angluin (1981) addressed the problem of inference of a deterministic finite automaton from given positive data and queries.

In this paper we are concerned with the problem of identifying pattern languages from both positive data and queries. The class of pattern languages was first introduced by Angluin (1980). Roughly speaking, a *pattern* p is a finite string over constant characters and variables. The *pattern language* $L(p)$ defined by a pattern p is the set of strings over constant characters that can be obtained from the pattern p by substituting each variable in p by a constant string. (See Section 2 for a precise definition.) For example, $0x01x$ is a pattern with constants 0 and 1 and a variable x, and $L(0x01x) = \{00010, 01011, 0000100, 0010101, \ldots\}$. Angluin (1980) studied the inference problem of pattern languages from positive data only. In particular, she found a polynomial-time algorithm that takes as input a finite set S of strings and outputs a one-variable pattern p such that $L(p)$ is a smallest pattern language defined by a one-variable pattern such that $S \subseteq L(p)$. (See also Jantke, 1984, and Ko and Hua, 1987, for the studies on the problem of inferring k-variable patterns from positive data for $k > 1$.)

In Angluin's study, the question is to identify from examples *one* pattern that satisfies the requirements, though potentially exponentially many solutions may exist even if the number of variables in the pattern and the length of the pattern are fixed. In a slightly different model, we may assume that there exists a predetermined unique pattern and the goal is to identify this pattern precisely. Then, it is natural to consider the inference method by queries. For convenience, the problem may be viewed as a two-person game. The first player A attempts to identify a k-variable pattern p that was chosen by (and is known only to) the second player B (for now we assume that k is predetermined). Initially, B provides A with a finite set S of strings in $L(p)$. Then B acts as an oracle, anwering questions of whether $s \in L(p)$ for each s queried by A. For each pattern p, let $|p|$ denote its length. It is easy to identify p by quering "$s \in L(p)$?" for all constant strings s of length $|p|$ (Corollary 2.5) and thus A can always identify the pattern by $2^{|p|}$ queries. Therefore, we say that player A wins the game, or the pat-

tern p is *polynomially inferable* from the initial sample S, if A can identify p by making only $\phi(|p|)$ many queries to B for some fixed polynomial ϕ.

A general goal of the study of inductive inference by the presentation of both examples and queries is to give a classification of the initial sample according to the polynomial inferability of the associated inference problem. In other words, we want to identify the relationships between the contents of the sample and the complexity of the inference problem. For instance, in the study of the inference of regular languages from psotive data and queries, Angluin (1981) identified that the class of initial examples that enable the polynomial inferability of finite automata is exactly the class of live-complete sets for finite automata. In our study, we give simple sufficient conditions on the initial sample S for the polynomial inferability of the patterns. From the point of view of a two-person game, a condition on initial samples is (*polynomially*) *sufficient* if A has a winning strategy that can always identify a pattern p with $\phi(|p|)$ queries for some polynomial ϕ, whenever the initial sample S prepared by B satisfies this condition. The sufficient condition on the initial samples usually involves not only simple properties of the example strings but also the interrelations between the example strings and the pattern.

Consider, for example, patterns with one variable x. We obtain the following sufficient conditions:

(i) The size of the sample S (the total number of bits to represent S) is bounded by a polynomial in $|p|$.

(ii) S contains two strings w_0 and w_1 which are obtained from p by substituting the variable x by two *incompatible* strings u_0 and u_1, respectively. (Two strings u_0 and u_1 are *incompatible* if neither is both a prefix and a suffix of the other.)

For instance, a simple sufficient sample is created by substituting just two strings u and v for x, where $|u| = |v|$ and $u \neq v$. Condition (i) is required so that if a "good" sample of size two is contained in a larger sample, it can always be found in polynomial time. To demonstrate that the condition is also necessary, we also show that if player B does not have to follow condition (ii) in the preparation of the initial sample S, then for any two polynomials ϕ and ψ, B can find a pattern p that cannot be identified within $\phi(|p|)$ many queries, though the sample given by B contains $\psi(|p|)$ words. The sufficiency of these simple examples is particularly interesting considering the fact that there is no polynomial time algorithm that finds patterns using queries alone (without any examples).

For the patterns with more than one variable, the generalized sufficient conditions have also been found. For instance, assume that p contains k variables $x_1, x_2, \ldots, x_k$. Denote by $p[t_i/x_i]_{i=1}^{k}$ the string obtained from p by

sustituting x_i by t_i for $i = 1,..., k$, and let $w(i)$ denote the ith bit of a string w. Then, the following condition on S is sufficient: there exist k pairs of incompatible strings $u_{i,0}, u_{i,1}$, $i = 1,..., k$, such that for all $w \in \{0, 1\}^k$, $p[u_{i,w(i)}/x_i]_{i=1}^k \in S$. (I.e., S contains 2^k strings which are obtained from all combinations of substitutions of x_i by $u_{i,0}$ or $u_{i,1}$.) Furthermore, we demonstrate an example in which p has two variables and S contains $2^{|p|}$ many examples but player A still needs to make an exponential number of queries to identify p.

We give, in Section 2, basic notation and properties about pattern languages and a more precise definition of "polynomially sufficient conditions." The inference problems of one-variable and k-variable, $k > 1$, patterns are studied in Sections 3 and 4, respectively. Section 5 contains some concluding remarks.

2. Definitions

Let Σ be a finite alphabet. Then Σ^* denotes the set of all finite strings over Σ, and $\Sigma^+ = \Sigma^* - \{\textit{the empty string}\}$. Let X be a countable set of symbols disjoint from Σ. A *pattern* over (Σ, X) is a finite string in $(\Sigma \cup X)^+$. The elements in Σ are called *constants* while the elements in X are called *variables*. In the rest of the paper, we assume that $\Sigma = \{0, 1\}$, and $X = \{x_1, x_2,...\}$. When there is only one variable, we write x for x_1. We often write u, v, w, y, z (with or without substripts) for elements of Σ^*, and p, q for patterns. Let S be a finite set. We write $\|S\|$ to denote the size of S. Let s be a string in $(\Sigma \cup X)^*$. We write $|s|$ to denote the length of s, and, for $i \leqslant s$, write $s(i)$ to denote the ith symbol in s. For i and j such that $1 \leqslant i \leqslant |s|$ and $1 \leqslant j \leqslant |s| + 1 - i$, we write substr$(s, i, j)$ to denote the substring t of s which begins at the position i and has length $|t| = j$. We say t is a prefix of s if $t = \text{substr}(s, 1, |t|)$, and t is a suffix of s if $t = \text{substr}(s, |s| + 1 - |t|, |t|)$. Let $s \in (\Sigma \cup X)^*$ and $a \in \Sigma \cup X$, we write $\#_a s$ to denote the number of occurrences of a in s, and write first(a, s) to denote the position of the first occurrence of a in s. (first$(a, s) = 0$ if a does not occur in s.) A *substitution* over $(\Sigma \cup X)$ is a mapping of X into Σ^+. We extend a substitution f to $(\Sigma \cup X)^*$ in the natural way and write $p[t_1/x_1, t_2/x_2,..., t_k/x_k]$, or $p[t_i/x_i]_{i=1}^k$ for short, to denote the string w obtained from p by the substitution f that maps each x_i to t_i, $i = 1,..., k$. For example, if $p = x_1 x_2 x_2$, then $p[00/x_1, 10/x_2] = 001010$.

Let p be a pattern over $(\{0, 1\}, X)$. We let var(p) denote the number of distinct variables occuring in p. The *pattern language* $L(p)$ defined by the pattern p is $\{ p[t_i/x_i]_{i=1}^k \mid k = \text{var}(p), t_i \in \{0, 1\}^+, i = 1,..., k \}$. For the sake of simplicity, we assume that equivalent patterns (i.e., those that can be obtained from each other by renaming the variables) have only one

canonical representation. Namely, the variables in a k-variable pattern are always $x_1, x_2, \ldots, x_k$ and if $1 \leqslant m < n \leqslant k$, then $\text{first}(x_m, p) < \text{first}(x_n, p)$. We write Π_k to denote the set of all patterns with exactly k variables, and let $\Pi = \bigcup_{k=1}^{\infty} \Pi_k$. Using the above defined canonical representation of patterns, the term "a unique pattern" is meaningful as justified by the following proposition.

PROPOSITION 2.1 (Angluin, 1980). *Let p and q be patterns in Π_k for some $k \geqslant 0$. Then $p = q$ iff $L(p) = L(q)$.*

Our model of computation is the oracle Turing machine (oracle TM) (Hopcroft and Ullman, 1979). The oracle TM tries to identify a pattern p from the input of a finite set S of examples in $L(p)$ and the oracle $L(p)$. Such an oracle TM is called an *inference oracle* TM. The complexity of an inference oracle TM is measured by both the run time of the machine and the number of queries it makes. In order to give a reasonable definition of the notion of *polynomial inferability* by inference oracle TMs, we first state some simple facts about identifying patterns by inference oracle TMs. For each $n \geqslant 0$, let $\{0,1\}^n = \{w \in \{0,1\}^* \mid |w| = n\}$, and for each $p \in \Pi$ and $n \geqslant |p|$, let $L(p)^{=n} = \{w \in L(p) \mid |w| = n\}$.

LEMMA 2.2. *Let $p, q \in \Pi_k$ have the same length n. Then $p = q$ iff $L(p)^{=n} = L(q)^{=n}$.*

Proof. The forward direction is immediate. For the backward direction, assume that $p \neq q$. Let i be the least integer such that $p(i) \neq q(i)$, and consider the following cases:

Case 1. Both $p(i)$ and $q(i)$ are constants. Then, $w = p[1/x_j]_{j=1}^{k} \in L(p)^{=n} - L(q)$ because $w(i) = p(i) \neq q(i)$, and $u(i) = q(i)$ for all $u \in L(q)^{=n}$.

Case 2. $p(i)$ is a constant and $q(i)$ is a variable. Assume, without loss of generality, $p(i) = 0$. Then, $w = q[1/x_l]_{l=1}^{k} \in L(q)^{=n} - L(p)$, because $w(i) = 1 \neq p(i)$.

Case 3. $q(i)$ is a constant and $p(i)$ is a variable. Symmetric to Case 2.

Case 4. Both $p(i)$ and $q(i)$ are variables. Assume that $p(i) = x_s$, $q(i) = x_t$ and $s < t$. From our definition of the canonical representation of the patterns, $m = \text{first}(x_s, q) < i$. Since i is the least integer such that $p(i) \neq q(i)$, we must have $p(m) = q(m) = x_s$). Now, $w = q[1/x_t, 0/x_j]_{j=1, j \neq t}^{k}$ is in $L(q)^{=n} - L(p)$, because $w(m) = 0$ and $w(i) = 1$ but $u(m) = u(i)$ for all $u \in L(p)^{=n}$. ∎

LEMMA 2.3. *Let $p, q \in \Pi_k$ have the same length n. Then, $\|L(p)^{=n}\| = 2^k$, and $L(p) \subseteq L(q)$ implies $p = q$.*

Proof. First, every $w \in L(p)^{=n}$ is obtained by a substitution $f(x_i) = t_i$, $t_i \in \{0, 1\}$, $i = 1,..., k$. There are only 2^k many such substitutions, and so $\|L(p)^{=n}\| \leqslant 2^k$. Also, two different substitutions always give two different words. So, $\|L(p)^{=n}\| = 2^k$. Now, if $L(p) \subseteq L(q)$, then $L(p)^{=n} = L(q)^{=n}$ because both sets have size 2^k. Therefore, $p = q$ follows from Lemma 2.2. ∎

COROLLARY 2.4. *There is an oracle* TM M *such that for each input* $p \in \Pi_k$, $k \geqslant 1$ *and each oracle* $L(q)$ *of some* $q \in \Pi_k$ *with* $|q| \geqslant |p|$, M *outputs "yes" iff* $p = q$, *and* M *makes only* 2^k *queries each of length* $|p|$.

Proof. Let $|p| = n$. The machine enumerates $L(p)^{=n}$ and queries all $w \in L(p)^{=n}$. It outputs "yes" iff all queries are answered "yes" by the oracle. We note that if $|q| = n$ then, by Lemma 2.2, $L(p)^{=n} = L(q)^{=n}$ iff $p = q$. So, M works correctly. If $|q| > n$ then $L(q)^{=n}$ is empty and the algorithm will output "no" correctly. ∎

COROLLARY 2.5. *There is an inference oracle* TM M *such that for each input* k *and each oracle* $L(p)$ *with* $\text{var}(p) = k$, M *outputs* p *by making* $2^{|p|+1} - 1$ *queries.*

Proof. The machine M repeatedly guesses $|p| = n$ for $n = 1, 2,...,$ and performs the following for each n. It enumerates all 2^n strings in $\{0, 1\}^n$ and queries each of them. If all the answers are "no" then n is too small and the machine proceeds to the next guess. As soon as $n = |p|$, some queries will be answered positively. In fact, there are exactly 2^k "yes" answers. Now M can easily determine the pattern p from these answers. Namely, define an equivalence relation R on $\{i \mid 1 \leqslant i \leqslant n\}$ by iRj iff for all queries s answered "yes," $s(i) = s(j)$. Then, each equivalence class defines either all the occurrences of a variable or a constant. More precisely, let $S = L(p)^{=n}$ (as obtained using queries) and, for all i, let $f(i) = \|\{s \in S \mid s(i) = 1\}\|$. Then, we get, for all i, $p(i) = 0$ if $f(i) = 0$ and $p(i) = 1$ if $f(i) = 2^k$. All the remaining i's have $f(i) = 2^{k-1}$ and form k equivalence classes. Let E_j, $1 \leqslant j \leqslant k$, be the jth equivalence class, in the canonical order. Then $p(i) = x_j$ iff $i \in E_j$. ∎

In the following, we give a precise definition of polynomial inferability of an inference oracle TM.

DEFINITION 2.1. A *presenting function* is a function f which maps each pattern $p \in \Pi$ to a finite subset of $L(p)$; The set $f(p)$ is called a sample for $L(p)$.

DEFINITION 2.2. Let $k \in N$ and C be a class of presenting functions. We

say that C is (*polynomailly*) *sufficient* for Π_k if there exist an oracle TM M and a polynomial ϕ such that for each $f \in C$ and each $p \in \Pi_k$,

(i) $M^{L(p)}(f(p)) = p$;

(ii) the computation of $M^{L(p)}(f(p))$ makes at most $\phi(|p|)$ queries; and

(iii) the computation of $M^{L(p)}(f(p))$ takes at most $\phi(|f(p)|)$ moves.

The presenting function is intended to formalize the interrelation between the sample S and the pattern p. From the two-person game point of view, a presenting function represents a method (not necessarily computable) which player B applies to prepare its sample S. If for some sample preparation method f, the player A has a winning strategy to identify each pattern by $\phi(|p|)$ queries and in total time $\phi(|f(p)|)$, for some fixed polynomial ϕ, then we say that such a method is sufficient.

Note that conditions (ii) and (iii) distinguish the *problem size* (the length of p) from the *input size* (the size of the sample S).

3. Inference of Patterns with One Variable

We first concentrate on a natural class of presenting functions: presenting functions which create the sample by substituting fixed words from $\{0, 1\}^+$ for the variable x in the pattern under consideration. In Theorems 3.4 and 3.5 we show that even when the sample size is limited to two, there is a very clear distinction between the sufficient and insufficient subclasses. Before we state our theorems, we first establish some basic properties about one-variable patterns.

Definition 3.1. We say that two words $u, v \in \{0, 1\}^+$ are *compatible* and denote by $\mathrm{com}(u, v)$, if either u or v is both a prefix and a suffix of the other. Otherwise, we say that u and v are *incompatible*.

Lemma 3.1. *Let u and v be two words in $\{0, 1\}^+$ with the property that for some $a > 0$, and $y, z \in \{0, 1\}^*$, $u^a v y = v z$. Then, there exist an integer $k \geqslant 0$ and a prefix u_1 of u such that $v = u^k u_1$.*

Proof. Let k and r satisfy $|v| = |u| \cdot k + r$ and $0 \leqslant r < |u|$. If $k = 0$, then $|v| < |u|$ and the lemma is trivial. So, assume that $k > 0$. Then, $u^a v y = v z$ implies that for some $b \geqslant 1$, u^b is a prefix of v. Let b be the greatest such integer. We consider two cases:

Case 1. If $b < k$, then write $v = u^b w$ for some $w \in \{0, 1\}^*$. We have $vz = u^a v y = u^a u^b w y = u^{a+b} w y$, in contradiction to the assumption that b is the greatest integer such that $v = u^b w$ for some $w \in \{0, 1\}^*$.

Case 2. If $b=k$ then, similar to Case 1, we have $vz=u^{a+b}wy$ for some $w\in\{0,1\}^*$. Since $a+b>b=k$, the desired result follows immediately. ▮

LEMMA 3.2. *Assume that* $u, v\in\{0,1\}^+$ *have the property that* $yvu^a=zv$ *for some* $a>0$ *and* $y, z\in\{0,1\}^*$. *Then* $v=u_2u^k$ *for some* $k\geqslant 0$ *and some suffix* u_2 *of* u.

Proof. Symmetric to the proof of Lemma 3.1. ▮

LEMMA 3.3. *Let* u *and* v *be two incompatible words, and* p *and* q *two one-variable patterns of the same length. If* $p[u/x]=q[u/x]$ *and* $p[v/x]=q[v/x]$ *then* $p=q$.

Proof. Assume, without loss of generality, that $|u|\leqslant|v|$. Then, there are two cases: u is not a prefix of v, or u is not a suffix of v.

First consider the case that u is not a prefix of v. We prove the lemma for this case by induction on $|p|$. The case of $|p|=|q|=1$ is trivial. For $|p|=|q|=n>1$, assume by contradiction that $p\neq q$. If the leftmost symbols of p and q are the same, the reduction to the case of $|p|=n-1$ is trivial. Otherwise, assume $\text{first}(x,p)=i>1$ and $\text{first}(x,q)=1$. Then, $p[u/x]=q[u/x]$ and $p[v/x]=q[v/x]$ imply $wuy=uz$ and $wvy'=vz'$ for some $w\in\{0,1\}^+$ and $y, y', z, z'\in\{0,1\}^*$. From Lemma 3.1, there exist integers k, j and two prefixes w_1, w_2 of w such that $u=w^kw_1$ and $v=w^jw_2$. Since $|u|\leqslant|v|$ and both w_1 and w_2 are prefixes of w, we get that u is a prefix of v, and thus a contradiction.

For the second case that u is not a suffix of v, the proof is symmetric using Lemma 3.2 instead of 3.1. ▮

Theorems 3.4 and 3.5 below provide necessary and sufficient conditions for sufficiency for a particular class of presenting functions: presenting functions which create the sample by substituting fixed words from $\{0,1\}^+$ for the variable x in the pattern under consideration. We will then show that Theorem 3.4 can be extended to cover a much larger class of presenting functions. On the other hand, the counterexample for the necessity of the condition proved in Theorem 3.5 cannot be extended in the same manner.

THEOREM 3.4. *Let* C_0 *be the class of all presenting functions such that for each* $f\in C_0$ *there exist two incompatible words* u *and* v *such that for all* $p\in\Pi_1$, $f(p)=\{p[u/x], p[v/x]\}$. *Then* C_0 *is sufficient for* Π_1.

Proof. The following algorithm identifies p.

procedure *inferpattern* (w_1, w_2);
{w_1 and w_2 are the sample; $plen=|p|$}

```
begin
  m := min{|w1|, |w2|};
  for plen := 1 to m do begin
    {As in Corollary 2.4, no pattern will be found if plen is less than the
    real length of the mystery pattern}
    for numx := 1 to plen do begin
      {guessing the number of occurrences of x in p}
      ulen := (|w1| - (plen - numx))/numx;
      vlen := (|w2| - (plen - numx))/numx;
      {compute |u| and |v|}
      {if ulen or vlen is not an integer, reject this numx}
      for i := 1 to plen do begin
        {guessing first(x, p)}
        u := substr(w1, i, ulen);
        v := substr(w2, i, vlen);
        if not com(u, v) then begin
          p := findpattern (w1, w2, u, v);
          if p ≠ λ then if confirm(p) then print (p) and halt
        end
      end
    end
  end;
  print(λ)
end;
```

In the above algorithm, confirm(p) is a function that returns **true** iff p is the right pattern. Corollary 2.4 shows that confirm(p) makes only two queries and runs in linear time.

Also, *findpattern* (w_1, w_2, u, v) is a function that, assuming *not* com(u, v), returns a pattern p such that $w_1 = p[u/x]$ and $w_2 = p[v/x]$ if such a p exists, and returns λ otherwise. We note that Lemma 3.3 assures the existence of *at most* one such pattern p. It is interesting to observe that p can be found in linear time if it exists. For instance, assume, without loss of generality, that $|u| \leqslant |v|$, and if u is not a prefix of v, then, we proceed from left to right and compute

$$d = \min\{i \mid i \leqslant |u|, u(i) \neq v(i)\},$$

and

$$D = \min\{i \mid i \leqslant |w_1|, w_1(i) \neq w_2(i)\}.$$

Then, $D - d + 1$ determines the next position where u and v in w_1 and w_2, respectively (and thus the next position of x in p), are expected. If u and v are actually found in those locations, then the leading constants and

x are appended to the pattern under construction, and w_1 and w_2 are reduced to the yet unexplored tail, and the above process repeats. The algorithm will determine that p does not exist when either u or v is not found where expected, or other length problems arise when matching w_1 and w_2. ▮

We have shown in the above that there is an algorithm that can identify the desired pattern if it is provided with an initial sample created by any presenting function that uses two fixed, incompatible words for substitution. On the other hand, the next theorem shows that if a presenting function f uses two fixed, compatible words for substitution, then $\{f\}$ is insufficient for Π_1.

THEOREM 3.5. *Let f be a presenting function such that there exist $u, v \in \{0, 1\}^+$, com(u, v), such that for all $p \in \Pi_1$, $f(p) = \{p[u/x], p[v/x]\}$. Then $\{f\}$ is insufficient for Π_1.*

Proof. The technique used in this proof is the "adversary argument." Since it will be used again in the k-variable case, $k > 1$, we first describe its general framework.

Let us again view this inference problem as a two-person game. Player A is the oracle TM and player B is the oracle. We let player B assume the role of the adversary. B plays the game according to the rule that all answers must be "consistent," but tries to present a worst case to his/her opponent. In other words, player B does not fix the chosen pattern, but keeps track of the set of all patterns which are consistent with the examples and previous answers to queries.

More precisely, B defines the *search space*, a set H_i which contains (not necessarily all) patterns that agree with all the initial examples and the first i queries. After the $(i+1)$th query "Is $w \in L(p)$?," B creates two sets:

$$G_i^Y(w) = \{p \in H_i \mid w \in L(p)\},$$

and

$$G_i^N(w) = \{p \in H_i \mid w \notin L(p)\} = H_i - G_i^Y(w)$$

and responds with "yes" iff $\|G_i^Y(w)\| \geqslant \|G_i^N(w)\|$. H_i is then set to either $G_i^Y(w)$ or $G_i^N(w)$, depending on the above answer. Thus B is maintaining the search space as large as possible within the consistency requirement.

To ensure that player A cannot reduce the size of the search space to one with polynomially many queries, the following are also required:

(i) $|p| = r$ is fixed in the search space;

(ii) there is a set $S_r \subseteq \{0, 1\}^*$ such that $\|S_r\| \leqslant \phi(r)$ and $(\forall s \in S_r)$

($|s| \leqslant \phi(r)$) for some polynomial ϕ, and $\|H_0\| \geqslant 2^{cr}$ for some $c > 0$, where $H_0 = \{ p \mid |p| = r,\ S_r \subseteq L(p) \}$; and

(iii) there exists a polynomial θ such that for all $w \in \{0, 1\}^*$ and all $i \geqslant 0$, $\min\{ \|G_i^Y(w)\|, \|G_i^N(w)\| \} \leqslant \theta(r)$.

Since after each query, the size of the search space is reduced by at most $\theta(r)$, player A can eliminate at most $\psi(r) \cdot \theta(r)$ patterns from the search space after $\psi(r)$ queries. Hence, for r such that $2^{cr} > \psi(r) \cdot \theta(r) + 1$, the remaining search space contains at least two patterns.

Several lemmas are needed to complete the proof of Theorem 3.5.

LEMMA 3.6 (cf. Lothaire, 1983). *Let $u, v \in \{0, 1\}^+$ commute ($uv = vu$). Then there exist $t \in \{0, 1\}^+$ and integers $a, b > 0$ such that $u = t^a$ and $v = t^b$.*

Proof. Assume, without loss of generality, that $|u| \leqslant |v|$. We prove it by induction on $|v|$. First, $|v| = 1$ implies $u = v$, and so $t = u$ and $a = b = 1$ suffice.

Assume the correctness for $|v| < n$, and examine the case $|v| = n$. If $|u| = |v|$, then again $t = u$ and $a = b = 1$ suffice. If $|u| < |v|$, then, by Lemma 3.1, there exist u_1, a prefix of u such that $v = u^k u_1$ for some integer $k > 0$. Now, $u^{k+1} u_1 = uv = vu = u^k u_1 u$ implies $uu_1 = u_1 u$. By the inductive hypothesis, there exist $t \in \{0, 1\}^+$ and integers a, c such that $u = t^a$ and $u_1 = t^c$. Substituting $b = ak + c$, we get $v = t^b$. ∎

The following lemma extends the concept of commutativity of words in Lemma 3.6 and basically states that if m_1 copies of u and m_2 copies of v can be ordered in more than one way yielding the same word w, then u and v commute. The statement of the lemma tries to avoid possible ambiguity.

LEMMA 3.7. *Let f and g be two finite mappings that define words w_1 and $w_2 \in \{0, 1\}^*$ as follows: $f, g: \{1, 2, \ldots, n\} \to \{u, v\}$, $\|f^{-1}(u)\| = \|g^{-1}(u)\|$, $f \neq g$, and $w_1 = f(1) \cdot f(2) \cdot \cdots \cdot f(n)$, $w_2 = g(1) \cdot g(2) \cdot \cdots \cdot g(n)$. Then $w_1 = w_2$ implies $uv = vu$.*

Proof. Assume, without loss of generality, that $|u| \leqslant |v|$. We again prove it by induction on n. For $n = 2$, $f \neq g$ implies $uv = vu$. Assume correctness for the cases $n \leqslant n_0$, and consider for $n = n_0 + 1$.

The cases where $f(1) = g(1)$ or $f(n_0 + 1) = g(n_0 + 1)$ can be easily reduced to the case of $n = n_0$. If $f(1) \neq g(1)$ and $f(n_0 + 1) \neq g(n_0 + 1)$, then $w_1 = w_2$ implies $u^a v y_1 = v z_1$ and $y_2 v u^b = z_2 v$ for some $a, b > 0$ and $y_1, y_2, z_1, z_2 \in \{0, 1\}^*$. By Lemmas 3.1 and 3.2 $(\exists k > 0)$ $(\exists u_1, u_2, u_3, u_4 \in \{0, 1\}^*)$ [$u = u_1 u_3 = u_4 u_2$, and $v = u^k u_1 = u_2 u^k$]. We have $|u_1| = |u_2|$, and hence $u_2 (u_1 u_3)^k = u_2 u^k = v = u^k u_1 = (u_1 u_3)^k u_1$. This implies $u_1 = u_2$. Since $k > 0$, $u^k u_1 = u_1 u^k$ implies $u (u_1 u_3)^{k-1} u_1 = u_1 u^k$, which

implies, by comparison of the first $|uu_1|$ symbols, that $uu_1 = u_1 u$. Hence $uv = u^{k+1}u_1 = u^k u_1 u = vu$. ∎

LEMMA 3.8. *Let $u, v \in \{0,1\}^+$. Then, u and v are compatible iff there exist t, $y \in \{0,1\}^*$ and $a, b > 0$ such that $uy = t^a$ and $vy = t^b$.*

Proof. Assume, without loss of generality, that $|u| \leqslant |v|$. Then, com-(u, v) implies that there exist $z, z' \in \{0,1\}^*$ such that $v = uz' = zu$. Applying Lemma 3.1 to z and u, we get that for some z_1 and z_2, $z = z_1 z_2$ and $u = z^k z_1$ for some $k \geqslant 0$. Hence, $v = zu = z^{k+1}z_1$. Choosing $y = z_2$, $t = z$, $a = k+1$, and $b = k+2$, we check that $uy = z^k z_1 z_2 = z^{k+1} = t^a$, and $vy = z^{k+1}z_1 z_2 = t^b$.

Conversely, assume $uy = t^a$ and $vy = t^b$ for some $y, t \in \{0,1\}^*$ and $a, b > 0$. Then $|u| \leqslant |v|$ implies that u is a prefix of v. Furthermore, since y is a suffix of t^a, we have $u = t^c t_1$ and $v = t^d t_1$ for some prefix t_1 of t. This implies that u is also a suffix of v. ∎

We now continue the proof of Theorem 3.5. Since f creates the sample using fixed substitution words u, v that satisfy com(u, v), there exist, by Lemma 3.8, t, $y \in \{0,1\}^*$, a, $b > 0$, such that $uy = t^a$ and $vy = t^b$. Let n be any even integer. Let $H_{0,n} = \{ p \in \Pi_1 \mid p \in (xy \cup uy)^n,\ \#_x p = n/2 \}$, and $r = n(|y| + |u|/2 + 1/2)$.

The following facts are observed:

(i) For all $p \in H_{0,n}$, $|p| = r$.

(ii) For all $p \in H_{0,n}$, $p[u/x] = t^{an}$ and $p[v/x] = t^{(a+b)n/2}$.

(iii) For all $p \in H_{0,n}$, $f(p) = \{t^{an}, t^{(a+b)n/2}\}$.

(iv) If $q_1, q_2 \in H_{0,n}$, $q_1 \neq q_2$ and $q_1[s/x] = q_2[s/x]$ for some $s \in \{0,1\}^+$, then there exist $t_1 \in \{0,1\}^*$ and $c, d > 0$ such that $sy = t_1^c$ and $uy = t_1^d$ (by Lemmas 3.6 and 3.7).

(v) If $s, t_1 \in \{0,1\}^+$ and $c, d > 0$ satisfy $sy = t_1^c$ and $uy = t_1^d$, then for all $q_1, q_2 \in H_{0,n}$, $q_1[s/x] = q_2[s/x]$.

(vi) If $w = q_1[s/x] = q_2[s/x]$ for some $q_1, q_2 \in H_{0,n}, q_1 \neq q_2$, and some $s \in \{0,1\}^+$, then $w \in L(p)$ for all $p \in H_{0,n}$.

(vii)

$$\|H_{0,n}\| = \binom{n}{n/2} = \frac{n!}{(n/2)!^2} = \frac{\prod_{i=n/2+1}^{n} i}{\prod_{i=1}^{n/2} i} = \prod_{i=1}^{n/2} \frac{i + n/2}{i} \geqslant 2^{n/2}.$$

Now the theorem follows from the following claim: Let $G_{0,n}^Y(w)$ be defined for every $w \in \{0,1\}^+$ as $G_{0,n}^Y(w) = \{ p \in H_{0,n} \mid w \in L(p) \}$. Then, $\| g_{0,n}^Y(w) \| \leqslant r$, or $G_{0,n}^Y(w) = H_{0,n}$.

Proof of claim. Let $G^Y_{0,n}(w) = \{p_1, p_2, \ldots, p_m\} \subseteq H_{0,n}$. Let $s_i \in \{0,1\}^+$ satisfy $w = p_i[s_i/x]$, $1 \leqslant i \leqslant m$. If for some $i \neq j$, $s_i = s_j$, then, by (vi) above, $G^Y_{0,n}(w) = H_{0,n}$. Otherwise, if all s_i, $1 \leqslant i \leqslant m$, are different, since they are all continuous subwords of w and are of the same length, they must differ by their starting positions in w. Since at most r constants may precede the first occurrence of x in any $p \in H_{0,n}$, we get that $m = \|G^Y_{0,n}(w)\| \leqslant r$.

Now, for any polynomial ψ, let n be large enough to satisfy $2^{n/2} > \psi(r) \cdot r$, and $S_r = \{t^{an}, t^{(a+b)n/2}\}$. (Note that $r = c \cdot n$ for some constant c.) Then, player A can reduce, for each query w, the size of the search space by at most $\|G^Y_{i,n}(w)\| = \|G^Y_{0,n}(w) \cap H_{i,n}\| \leqslant r$, and cannot identify a unique pattern in $\psi(r)$ queries. This concludes the proof of Theorem 3.5. ∎

We have identified, in Theorems 3.4 and 3.5, a necessary and sufficient condition for the property of *polynomial sufficiency for* Π_1 for presenting functions that are based on two fixed substitution words. The proofs genealize to the following classes of presenting functions. Let C be the class of presenting functions f such that there is a polynomial ϕ that bounds both the sample size and the sizes of individual examples. That is, $C = \{f \mid f$ is a presenting function and $(\exists$ polynomial $\phi)$ $(\forall p \in \Pi_1)$ $(\forall w \in f(p))$ $[\|f(p)\| \leqslant \phi(|p|)$ and $|w| \leqslant \phi(|p|)]\}$.

COROLLARY 3.9. *Let* $C_1 = \{f \in C \mid (\forall p \in \Pi_1)$ $(\exists u, v \in \{0,1\}^+)$ $[$*not* com$(u, v)]$ *and* $p[u/x]$, $p[v/x] \in f(p)\}$. *Then* C_1 *is sufficient for* Π_1.

As stated before, this changing of the order of the quantifiers cannot be applied to Theorem 3.5. To see this, consider, for example, the presenting function f such that $f(p) = \{p[u/x], p[v/x]\}$, where u is always 1, and $v = 1^{|p|}01^{|p|}$. We have com(u, v), but identification is simple as the matching process can be anchored at points where a single zero is surrounded by long sequences of ones, in the longer word in the sample.

COROLLARY 3.10. *Let* $C_2 = \{f \in C \mid (\forall p \in \Pi_1)$ $(\exists w_1, w_2 \in f(p))\ |w_1| = |w_2|$ *and* $w_1 \neq w_2\}$. *Then* C_2 *is sufficient for* Π_1.

Remark. Note that C_2 is defined by a simple property on the samples instead of a property on the presenting functions.

COROLLARY 3.11. *Let* f *be a presenting function in* C *such that for infinitely many* $r \in N$, *all the patterns* $p \in \Pi_1$ *with* $|p| = r$ *satisfy* $f(p) = \{p[u/x], p[v/x]\}$ *for the same pair of compatible* u *and* v. *Then,* $\{f\}$ *is insufficient for* Π_1.

COROLLARY 3.12. *Let* $f \in C$ *be such that* $\|f(p)\| \leqslant 1$. *Then* $\{f\}$ *is insufficient for* Π_1.

COROLLARY 3.13. *For any polynomial ϕ, there is a presenting function f such that for infinitely many $p \in \Pi_1$, $\|f(p)\| \geq \phi(|p|)$ but $\{f\}$ is insufficient for Π_1.*

Proof. For any polynomial ϕ, and any even integer n with $2^{n/2} > \phi(n) \cdot n$, choose $H_{0,n} = \{p \in (1 \cup x)^n \mid \#_x p = n/2\}$, and $f(p) = \{1^{(k+1)n/2} \mid 1 \leq k \leq \phi(n)\}$. Then, it is easy to check that the argument used in Theorem 3.5 applies to here also. ∎

4. INFERENCE OF PATTERNS WITH MORE THAN ONE VARIABLE

It seems natural to expect that patterns with more than one variable are harder to identify than those with one variable. The results of this section give clear evidence to that effect. We need the following definition for the discussion of the results.

DEFINITION 4.1. We say that a presenting function f is *strongly insufficient* for Π_k, $k > 1$, if there exist a constant c and a polynomial ϕ such that

(i) $(\forall p \in \Pi_k) f(p) = S$ has the property that $\|S\| \geq 2^{c|p|}$, and $(\forall s \in S)\ |s| \leq \phi(|p|)$, and

(ii) for any polynomials ψ and θ, there is no oracle TM M that identifies all patterns from $f(p)$ (i.e., $(\forall p \in \Pi_k)\ M^{L(p)}(f(p)) = p$), and only makes at most $\psi(p)$ queries, each of length bounded by $\theta(|p|)$.

This definition is stronger than our original one as f must give a large sample that appears to be exhaustive and we allow the oracle TM to make an exponential number of nonquery moves. Note that there is no presenting function that is strongly insufficient for Π_1.

The main reults of this section may be summarized as follows:

(1) An insufficient class of presenting functions for Π_1 can be easily modified to an insufficient class for Π_k, for any $k > 1$.

(2) There are presenting functions which are *strongly insufficient* for Π_k, $k > 1$.

(3) The sufficient classes we have identified for the k-variable case with $k > 1$ are more complex than the sufficient classes for Π_1. Instead of a trivial extension, we observe, in these classes, subtle relationships between substitutions for different variables.

We first present insufficient presenting functions. Throughout this section, when considering Π_k, we assume that k is fixed and known a *priori* to

the identification algorithm. This assumption is further discussed at the end of the section.

THEOREM 4.1. *Let $k>1$. Let f be a presenting function such that, for every $p \in \Pi_k$, the sample S generated by $f(p)$ has the property that* $(\exists i \leqslant k)$ $(\forall s_1, s_2 \in S)$ $[s_1 = p[u_{1,j}/x_j]_{j=1}^k$ *and* $s_2 = p[u_{2,j}/x_j]_{j=1}^k$ *imply* $\mathrm{com}(u_{1,i}, u_{2,i})]$. *Then $\{f\}$ is insufficient for Π_k.*

Proof. A trivial extension of Theorem 3.5. ∎

THEOREM 4.2. *There exists a presenting function which is strongly insufficient for Π_2.*

Proof. The technique used here is almost identical to the one used in Theorem 3.5. We follow the notation established there. For any even integer $r>0$, let $H_{0,r} = \{ p \in \Pi_2 \mid p \in x_1(x_1 \cup x_2)^{r-1},\ \#_{x_1} p = \#_{x_2} p \}$, and define $S_r = \{ v^r \mid v \in \{0,1\}^r \}$. Then, we have $\|H_{0,r}\| \geqslant 2^{r/2-1}$ and, for all $s \in S_r$, $|s| = r^2$. Also, $S_r \subseteq \bigcap_{p \in H_{0,r}} L(p)$.

Let $G_0^Y(w) = \{ p \in H_{0,r} \mid w \in L(p) \}$, and θ be any polynomial. We claim that if $|w| \leqslant \theta(r)$ then either $\|G_0^Y(w)\| \leqslant r \cdot \theta(r)$, or $G_0^Y(w) = H_{0,r}$.

Proof of Claim. Let $G_0^Y(w) = \{ p_1, p_2, \ldots, p_m \} \subseteq H_{0,r}$. Assume that $m > r \cdot \theta(r)$. We want to show that $G_0^Y(w) = H_{0,r}$.

Let $u_{1,i}, u_{2,i} \in \{0,1\}^+$, $1 \leqslant i \leqslant m$, be words that make $w = p_i[u_{1,i}/x_1, u_{2,i}/x_2]$, and let $E_j = \{ p_i \in G_0^Y(w) \mid |u_{1,i}| = j \}$, $1 \leqslant j \leqslant \theta(r)$. Since $\|G_0^Y(w)\| = m > r \cdot \theta(r)$, there exists a j_0, $1 \leqslant j_0 \leqslant \theta(r)$ such that $\|E_{j_0}\| \geqslant r$. Let us examine the leftmost position of the variable x_2 in any $p \in H_{0,r}$. There are at most $r/2$ alternatives for this choice and, hence, there must be two patterns in E_{j_0} in which the leftmost positions of x_2 are identical. Without loss of generality, call them p_1 and p_2. Then, $|u_{1,1}| = |u_{1,2}| = j_0$ implies $|u_{2,1}| = |u_{2,2}|$ and it follows immediately that $u_{1,1} = u_{1,2}$ and $u_{2,1} = u_{2,2}$. In other words, the word w can be constructed from two different sequences of $u_{1,1}$ and $u_{2,1}$, each with $r/2$ many occurrences of $u_{1,1}$ and $r/2$ many occurrences of $u_{2,1}$. By Lemmas 3.6 and 3.7, there exists $t \in \{0,1\}^*$ such that $u_{1,1} = t^a$ and $u_{2,1} = t^b$ for some $a, b > 0$. Hence $w = t^{(a+b)r/2}$. However, for all $p \in H_{0,r}$, we can substitute t^a and t^b for x_1 and x_2 and get $p[t^a/x_1, t^b/x_2] = w$. This shows that $G_0^Y(w) = H_{0,r}$ and proves the claim.

We note that for any $i>0$, $G_i^Y(w) = G_0^Y(w) \cap H_{i,r}$ and $\min\{ \|G_i^Y(w)\|, \|G_i^N(w)\| \}$ is bounded by a polynomial $r \cdot \theta(r)$. Therefore, for any polynomials ψ and θ, if r is large enough to satisfy $2^{(r/2)-1} > r \cdot \psi(r) \cdot \theta(r)$, then $\psi(r)$ many queries can eliminate from the search space at most $r \cdot \psi(r) \cdot \theta(r)$ patterns and cannot uniquely identify the pattern p in question. This completes our proof. ∎

It may seem that part of the strong insufficiency in the above proof lies within the creation of the sample using the same substitution for x_1 and x_2. In the following we demonstrate a simple construction that does not have this constraint.

For any $r = 4m > 0$, let $H_{0,r} = \{ p \in \Pi_2 \mid p \in x_1 1(x_1 1 \cup 1x_2)^{r/2-1}, \#_{x_1} p = \#_{x_2} p \}$, and $S_r = \{(10v1)^{r/2} \mid v \in \{0,1\}^r\}$. Then, it is obvious that $\|H_{0,r}\| \geqslant 2^{r/4-1}$, and for each $w \in S_r$, $|w| = r(r+3)/2$. Furthermore, $S_r \subseteq \bigcap_{p \in H_{0,r}} L(p)$, because for each $p \in H_{0,r}$ and each $v \in \{0,1\}^r$, $p[10v/x_1, 0v1/x_2] = (10v1)^{r/2}$. Following the proof of Theorem 4.2, we can prove that such a setting gives a stongly insufficient presenting function. Notice that for each word $w \in S_r$ and each $p \in H_{0,r}$, the substitutions for x_1 and x_2 are $10v$ and $0v1$, respectively, and are distinct.

COROLLARY 4.3. *For each $k > 1$, there exists a presenting function which is strongly insufficient for Π_k.*

We now investigate presenting functions which are sufficient for Π_k, $k > 1$. In the following proofs, the extension from the two-variable case to the k-variable cases, with $k > 2$, is not necessarily obvious, so the general case is presented directly. We identify a simple sufficient condition, that is a generalization of the result on the one-variable case. Basically, a sufficient sample contains for each variable x_i, at least one pair of words which differ from each other by the substitution for x_i. In addition, such pairs of words must be effectively identifiable among many other examples.

DEFINITION 4.2. We say that two sequences $\langle u_1, u_2, \ldots, u_k \rangle$ and $\langle v_1, v_2, \ldots, v_k \rangle$ are *uniformly incompatible* if either, for all i, $1 \leqslant i \leqslant k$, u_i and v_i are not prefixes of each other, or, for all i, $1 \leqslant i \leqslant k$, u_i and v_i are not suffixes of each other.

THEOREM 4.4. *Let C_k, $k > 1$, be a class of presenting functions such that, for each $f \in C_k$, there is a polynomial ϕ such that for each $p \in \Pi_k$, there exist $u_i, v_i \in \{0,1\}^+$, $1 \leqslant i \leqslant k$, with the following properties:*

(i) *the sequences $\langle u_i \rangle_{i=1}^k$ and $\langle v_i \rangle_{i=1}^k$ are uniformly incompatible;*

(ii) $f(p) = S = \{ p[u_j/x_j]_{j=1}^k \} \cup \{ p[v_i/x_i, u_j/x_j]_{j=1, j \neq i}^k \mid 1 \leqslant i \leqslant k \}$;

(iii) $|u_i|, |v_i| \leqslant \phi(|p|)$ *for all $i \leqslant k$.*

Then, C_k is sufficient for Π_k.

Remark. Basically, the sample S contains a basis word $p[u_j/x_j]_{j=1}^k$, and k additional words which differ from the basis word only by a single substitution.

Proof. Without loss of generality, we assume that for each i, $1 \leqslant i \leqslant k$, u_i and v_i are not prefixes of each other. The following algorithm, using oracle $L(p)$, identifies the pattern p from $f(p)$ and satisfies the requirements for polynomial sufficiency. (The algorithm can be easily extended so that if the assumption that u_i and v_i are not prefixes of each other leads to failure, then the symmetric case where u_i and v_i are not suffixes of each other is tried.)

Repeat Steps 1 through 6 for $|p| = 1, 2, \ldots$ **until** a pattern is found. As in Theorem 3.4, no pattern will be confirmed whose length is less than the length of the mystery pattern.

Step 1. *Identifying the pairs of words which differ by a single substitution.*

Build the sets $A_{i,0}$, $A_{i,1}$, for $1 \leqslant i \leqslant k$, as follows: Let $d = \min\{j \mid (\exists s_1, s_2 \in S) s_1(j) \neq s_2(j)\}$. Then, let

$$A_{1,0} = \{s \in S \mid s(d) = 0\},$$

and

$$A_{1,1} = \{s \in S \mid s(d) = 1\}.$$

Neither of $A_{1,0}$ and $A_{1,1}$ is empty. Moreover, by construction of S, one of them contains exactly one word, and the other contains the rest. For $i \geqslant 2$, let $A_{i-1,m}$ be the larger of $A_{i-1,0}$ and $A_{i-1,1}$. Define $d = \min\{j \mid (\exists s_1, s_2 \in A_{i-1,m}) s_1(j) \neq s_2(j)\}$, and let

$$A_{i,0} = \{s \in A_{i-1,m} \mid s(d) = 0\},$$

and

$$A_{i,1} = \{s \in A_{i-1,m} \mid s(d) = 1\}.$$

Since the leftmost occurrence of x_i in p is to the left of the leftmost occurrence of x_{i+1}, the above operation splits, by the substitution for x_i, the set S into two sets. One sets contains only the word in which v_i is substituted for x_i, and the other set contains all other words (in which u_i is substituted for x_i). (This can be done only because the u_i's and the v_i's are not prefixes of each other. In the symmetric case, when the u_i's and the v_i's are not suffixes of each other, the scanning of the words will proceed from right to left.)

Step 2. *Identifying the origin of each word in the sample.*

From Step 1, for each i, $1 \leqslant i \leqslant k-1$, one of $A_{i,0}$ and $A_{i,1}$ contains exactly one word. Call it s_i. For $i = k$, both $A_{k,0}$ and $A_{k,1}$ contain exactly

one word. Arbitrarily call them s_0 and s_k. (If the following Steps 3 through 6 fail, swap s_0 and s_k and repeat those steps.)

We may assume now $s_0 = p[u_j/x_j]_{j=1}^k$, and $s_i = p[v_i/x_i, u_j/x_j]_{j=1, j\neq i}^k$, $1 \leqslant i \leqslant k$.

Step 3. *Guessing substitutions.*

Each word s_i, $i > 0$, differs from s_0 only by the substitution for x_i. We guess the substitutions u_i and v_i, for all i, by creating a set of quadruples:

$$Q_i = \{\langle num_i, lenu_i, lenv_i, first_i\rangle \mid 1 \leqslant num_i \leqslant |p|, 1 \leqslant lenu_i \leqslant |s_0|,$$
$$1 \leqslant lenv_i \leqslant |s_i|, 1 \leqslant first_i \leqslant |s_0|\},$$

where num_i is a guess for $\#_{x_i} p$, $lenu_i$ and $lenv_i$ are guesses for $|u_i|$ and $|v_i|$, respectively, and $first_i$ is a guess for the leftmost position of the substitution for x_i in s_0. For each i, there are at most $|p|^4 \cdot \phi(|p|)^3$ quadruples.

Step 4. *Cleanup.*

For each $i > 0$, we treat s_0 and s_i as two words derived from a one-variable pattern $q_i = p[u_j/x_j]_{j=1, j\neq i}^k$. Then, by the argument of Theorem 3.4, we can determine which quadruples in Q_i "fit" s_0 and s_i. More precisely, for each quafruple $\langle num_i, lenu_i, lenv_i, first_i\rangle$, we get $u_i = \text{substr}(s_0, first_i, lenu_i)$ and $v_i = \text{substr}(s_i, first_i, lenu_i)$, and then use the procedure *findpattern* of the proof of Theorem 3.4 to determine whether (and how) the quadruple "fits" s_0 and s_i. If the quadruple does not fit s_0 and s_i (i.e., from the derived u_i and v_i, and words s_0 and s_i, we cannot find a one-variable pattern q_i), then we discard this quadruple from Q_i.

Step 5. *Combining quadruples.*

The remaining quadruples form up to $(|p|^4 \cdot \phi(|p|)^3)^k$ combinations. Each combination, as shown in Step 4, determines a guess of u_i and v_i, $1 \leqslant i \leqslant k$. This guess may or may not fit the sample. The following lemma provides an algorithm that determines whether there is a pattern which fits the guess, and uniquely identifies it if it exists.

LEMMA 4.5. *Given u_i, v_i, $1 \leqslant i \leqslant k$, and s_i, $0 \leqslant i \leqslant k$, there is at most one pattern $p \in \Pi_k$ satisfying $s_0 = p[u_j/x_j]_{j=1}^k$ and $s_i = p[v_i/x_i, u_j/x_j]_{j=1, j\neq i}^k$, $1 \leqslant i \leqslant k$.*

Proof. From Step 4, we can find, for each $i \leqslant k$, a word q_i over $\{0, 1, x_i\}^+$, such that if $p \in \Pi_k$ satisfies the above condition, then $q_i = p[u_j/x_j]_{j=1, j\neq i}^k$.

Now, the following derives p symbol by symbol:

$const = |s_0| - \sum_{i=1}^{k} |u_i| \cdot \#_{x_i} q_i$; (the number of constants in p}
$n := const + \sum_{i=1}^{k} \#_{x_i} q_i$; {$n = |p|$};
for $j := 1$ **to** n **do begin**
  if all $q_i(1)$ are identical
    then begin
      $p(j) := q_1(1)$;
      delete the first bit from all q_i
    end
  else if $q_i(1) = x_i$ for exactly one i
    then if for all $t \neq i$, u_i is a prefix of q_t
      then begin
        $p(j) := x_i$;
        delete the first x_i from q_i;
        for all $t \neq i$, delete the first u_i from q_t
      end
      else {not consistent} fail
    else {not consistent} fail
end;

This completes the proof of Lemma 4.5. ∎

Step 6. *Confirming/rejecting patterns.*

Using Corollary 2.4, we confirm or reject each pattern constructed in Step 5 by at most 2^k queries. The total number of queries, including the potential reiteration mentioned in Step 2, is $\leqslant 2^{k+1} \cdot |p|^{4k} \cdot \phi(|p|)^{3k}$ and is bounded by a polynomial in $|p|$ (with degree depending on k). Furthermore, the total number of computation steps is also bounded by a polynomial in $|p|$. ∎

It is not clear whether the requirement of uniform incompatibility of Theorem 4.4 can be simplified to "for all i, not com(u_i, v_i)."

COROLLARY 4.6. *If f is a presenting function such that there is a polynomial ϕ such that for each $p \in \Pi_k$, $k > 1$, $f(p) = S$ contains a subset S' of $k+1$ words with the property required in Theorem* 4.4, *and* $(\frac{\|s\|}{\|s'\|}) \leqslant \phi(|p|)$ *then $\{f\}$ is sufficient for Π_k.*

A special case of Corollary 4.6 is when S consists of 2^k words derived from a complete substitutions of pairs $\{u_i, v_i\}$, $1 \leqslant i \leqslant k$, for the pattern. More precisely, for each $k > 1$ and each j, $0 \leqslant j \leqslant 2^k - 1$, let $b(k, j)$ be the jth word in $\{0, 1\}^k$. (I.e., $|b(k, j)| = k$ and $b(k, j)$ is the k-bit binary representation of the integer j.)

COROLLARY 4.7. *Let f be a presenting function with the following property: For each $p \in \Pi_k$, $S = f(p)$ consists of 2^k words: $S = \{s_j \mid 0 \leqslant j \leqslant 2^k - 1\}$ such that $s_j = p[t_{i,j}/x_i]_{i=1}^k$, with $t_{i,j} = u_i$ if the ith bit of $b(k, j)$ is 0 and $t_{i,j} = v_i$ if the ith bit of $b(k, j)$ is 1, and the sequences $\langle u_i \rangle_{i=1}^k$ and $\langle v_i \rangle_{i=1}^k$ are uniformly incompatible. Then $\{f\}$ is sufficient for Π_k.*

Remark. A search using a direct application of the algorithm in Theorem 4.4 has a time complexity $\binom{2^k}{k+1}$ times that of the algorithm in Theorem 4.4. However, a much simpler solution is available. Apply the same technique as Step 1 of the algorithm in Theorem 4.4. Then, $\|A_{i,0}\| = \|A_{i,1}\|$, and we can create a complete binary tree of height k by splitting each nonsingleton set into two according to the substitution for the next variable. The desired set of $k+1$ words can then be easily picked up from the leaves (those numbered 0, 1, 2, 4,..., 2^{k-1}) of the tree.

In the preceding discussion, the number of variables, k, is fixed in the identification algorithms. The algorithms work only if this assumption regarding k is correct. It can be shown that this assumption cannot be relaxed, and the discovery of k cannot be included in the algorithm if the current definitions of inferability are to be retained. The main reason is the fact that (sufficient) samples for patterns in Π_k can be viewed also as (insufficient) samples for an unknown pattern of $k' > k$. Hence, an adversary can construct cases where for any claim of completion by the identification algorithm, he or she can claim that the mystery pattern was not identified. Testing an ever increasing k will not solve the problem, since no efficient termination conditions can be devised (obviously, k is bounded by the size of the smallest string in the sample). Another way to view this issue is to examine the "confirmation" process which is so crucial in the algorithm presented above. If k is fixed, any pattern in Π_k can be confirmed using at most 2^k queries. However, it is easy to construct patterns of $k' > k$ that will "pass" the confirmaktion test for Π_k.

5. CONCLUSION

We have studied the polynomial inferability of pattern languages from examples and queries. We identified sufficient conditions on the relationship between the pattern and the examples that guarantee the polynomial inferability of any pattern. Namely, the incompatibility of the substitution words provides a simple search algorithm that identifies the unknown pattern by a polynomial number of queries. We also demonstrated counterexamples that show the necessity of the incompatibility of the substitution words.

Our study falls into the general area of inductive inference theory (Angluin and Smith, 1983), and our approach is close, in spirit, to the algorithmic-complexity theory. In a more recent paper, Vailiant (1984) has studied the inductive inference of Boolean circuits from examples and queries. He has listed three general criteria for good learning machines: (1) The machines provably learn whole classes of concepts, and these classes can be characterized. (2) The classes of concepts are appropriate and non-trivial. (3) The computational process of the machines requires a polynomial number of steps. It is ready to verify that our study satisfies all three criteria. The main difference between our learning model and Valiant's is that we adopt the worst-case complexity measure in our polynomial analysis while he assumes that examples are given randomly and so, naturally, uses the average-case analysis.

A number of questions about the inductive inference of pattern languages remain open. In particular, a complete classification of presenting functions by their polynomial sufficiency appears to be an important but difficult problem. When the number of variables k is greater than 1, our sufficient condition (namely, uniform incompatibility) seems somewhat weak. However, matching strings which are not uniformly incompatible seems to be a very complex task. It is not clear whether samples constructed using fixed, incompatible, but not uniformly incompatible strings can be shown to be insufficient for some patterns. More understanding of the combinatorics of words may be helpful in this direction. Pattern languages have simple representations that, in general, admit simple inductive inference algorithms (cf. Angluin, 1980). It is desirable to extend this type of study to more complex formal languages and to explore the limit of polynomial inferability of these languages.

RECEIVED June 10, 1985; ACCEPTED June 3, 1986

REFERENCES

ANGLUIN, D. (1978), On the complexity of minimum inference of regular sets, *Inform. and Control* **39**, 337–350.

ANGLUIN, D. (1980), Finding patterns common to a set of strings, *J. Comput. System Sci.* **21**, 46–62.

ANGLUIN, D. (1981), A note on the number of queries needed to identify regular languages, *Inform. and Control* **51**, 76–87.

ANGLUIN, D., AND SMITH, C. H. (1983), Inductive inference: Theory and methods, *Comput. Surveys* **15**, 237–269.

GOLD, E. M. (1967), Language identification in the limit, *Inform. and Control* **10**, 447–474.

GOLD, E. M. (1972), System identification via state characterization, *Automatica* **8**, 621–636.

GOLD, E. M. (1978), Complexity of automaton identification from given data, *Inform. and Control* **37**, 302–320.

HOPCROFT, J. E., AND ULLMAN, J. D. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, Mas.

JANTKE, K. P. (1984), Polynomial time inference of general pattern languages, *in* "Proceedings, Symposium of Theoretical Aspects of Computer Science," Lecture Notes in Computer Science Vol. 166, pp. 314–325, Springer, New York/Berlin.

KO, K., AND HUA, C. (1987), A note on the two-variable pattern-finding problem, *J. Comput. System Sci.* **34**, 75–86.

LOTHAIRE, M. (1983), "Combinatorics on Words, Encyclopedia of Mathematics and its Applications," Vol. 17, Chap. 1, Addison–Wesley, Reading, Mas.

PAO, T. W., AND CARR, J. W., III (1978), A solution of the syntactical induction-inference problem for regular languages, *Comput. Lang.* **3**, 53–64.

VALIANT, L. G. (1984), A theory of the learnable, *Comm. ACM* **27**, 1134–1142.